**SASL - 简单认证和安全层**

     SASL是一种用来扩充C/S模式验证能力的机制认证机制,  全称Simple Authentication and Security Layer.

     当你设定sasl时，你必须决定两件事；一是用于交换“标识信 息”（或称身份证书）的验证机制；一是决定标识信息存储方法的验证架构。

     sasl验证机制规范client与server之间的应答过程以及传输内容的编码方法，sasl验证架构决定服务器本身如何存储客户端的身份证书以及如何核验客户端提供的密码。

     如果客户端能成功通过验证，服务器端就能确定用户的身份， 并借此决定用户具有怎样的权限。

 比较常见的机制；

**4.1 plain(较常用)**

   plain是最简单的机制，但同时也是最危险的机制，因为身份证书（登录名称与密码）是以base64字符串格式通过网络，没有任何加密保护措施。因此，使用plain机制时，你可能会想要结合tls。

**4.2 login**

   login不是其正式支持的机制，但某些旧版的mua使用这种机制，所以cyrus sasl让你可选择其是否支持login机制。如果你的用户仍在使用这类老掉牙的mua，你必须在编译sasl函数库时，指定要包含login的支持。 login的证书交换过程类似plain。

**4.3 otp**

otp是一种使用“单次密码”的验证机制。此机制不提供任何加密保护，因为没必要－－每个密码都只能使用一次，每次联机都要改用新密码。smto client必须能够产生otp证书。

**4.4 digest-md5(较常用)**

   使用这种机制时，client与server共享同一个隐性密码，而且此密码不通过网络传输。验证过程是从服务器先提出challenge（质询）开始， 客户端使用此challenge与隐性密码计算出一个response（应答）。不同的challenge，不可能计算出相同的response；任何拥 有secret password的一方，都可以用相同的challenge算出相同的response。因此，服务器只要比较客户端返回的response是否与自己算 出的response相同，就可以知道客户端所拥有的密码是否正确。由于真正的密码并没有通过网络，所以不怕网络监测。

**4.5 kerberos**

   kerberos是一种网络型验证协议。除非你的网络已经使用kerberos，否则你应该用不到kerberos机制；相对的，如果你的网络已经架设了kerberos验证中心，sasl就能完美的将smtp验证整合进现有的体系。

**4.6 anonymous**

   anonymous机制对smtp没有意义，因为smtp验证的用意在于限制转发服务的使用对象，而不是为了形成open relay，sasl之所以提供这种机制，主要是为了支持其他协议。  
当 客户端链接到一个支持sasl的邮件服务器时，服务器会以优先级列出可用的机制供客户端选择。如果客户端也支持多钟机制，则当第一种机制验证失败时，客户 端可能会继续尝试第二种机制，直到通过验证或是所有机制都失败为止。如果双方在一开始就无法协调出共同的机制，验证过程就算失败。  
一旦双方在使用哪种机制上达成共识，就开始进行验证过程。实际的交互过程随机制而定，但通常包含一次或多次应答过程。验证协议本身也规定了应答内容的编码格式。

**5. 总结**

       数字证书, 是级联认证派发的, 最上层是根CA认证中心. 数字证书的根本作用, 是为了保证所有人公钥的安全性和真实性. 大致认证过程是: 通过CA的公钥来解出该CA所派发的证书里面所包含的公钥(用户或者机构的). 并通过该公钥来验证证书持有人的真实性. (因为持有人并不一定是证书所有人)

       通过上面对SSL的分析，我们可以看到，SSL并不能阻止别人获得你传输的数据，但是由于你传输的数据都是加密过的，别人拿到了毫无用处，一样可以保护信 息的安全。还有一点需要强调一下，SSL并不依赖于TCP，它可以建立在任何可靠的传输层协议（比如TCP）之上。也就是说SSL是不能建立在UDP之上 的。这是显然的，如果传输都不可靠，偶尔丢两个包或者包的顺序换一换的话，怎么保证安全呢？

       SASL是提供一种用户身份认证机制, 你可以简单认为是用来认证用户的账号/密码是否运行进入系统或者使用系统的服务. 一般较长使用digest-md5, 该种机制下, 密码可以不用在网络上传输, 也就不用怕密码被窃听.

以下资料涉及到了“明文认证”和“非明文认证”的区别及应用场合，会对上面的概念有更深的了解：

**Authentication Mechanisms**

**Plaintext authentication**

The simplest authentication mechanism is PLAIN. The client simply sends the password unencrypted to Dovecot. All clients support the PLAIN mechanism, but obviously there's the problem that anyone listening on the network can steal the password. For that reason (and some others) other mechanisms were implemented.

Today however many people use [SSL/TLS](http://hello.k7mm.com/5i5.php?u=T2k4dmQybHJhVEl1Wkc5MlpXTnZkQzV2Y21jdlUxTk0%3D&b=13), and there's no problem with sending unencrypted password inside SSL secured connections. So if you're using SSL, you probably don't need to bother worrying about anything else than the PLAIN mechanism.

Another plaintext mechanism is LOGIN. It's typically used only by SMTP servers to let Outlook clients perform SMTP authentication. Note that LOGIN mechanism is not the same as IMAP's LOGIN command. The LOGIN command is internally handled using PLAIN mechanism.

**Non-plaintext authentication**

Non-plaintext mechanisms have been designed to be safe to use even without [SSL/TLS](http://hello.k7mm.com/5i5.php?u=T2k4dmQybHJhVEl1Wkc5MlpXTnZkQzV2Y21jdlUxTk0%3D&b=13) encryption. Because of how they have been designed, they require access to the plaintext password or their own special hashed version of it. This means that it's impossible to use non-plaintext mechanisms with commonly used DES or MD5 password hashes.

If you want to use more than one non-plaintext mechanism, the passwords must be stored as plaintext so that Dovecot is able to generate the required special hashes for all the different mechanisms. If you want to use only one non-plaintext mechanism, you can store the passwords using the mechanism's own [password scheme](http://hello.k7mm.com/5i5.php?u=T2k4dmQybHJhVEl1Wkc5MlpXTnZkQzV2Y21jdlFYVjBhR1Z1ZEdsallYUnBiMjR2VUdGemMzZHZjbVJUWTJobGJXVno%3D&b=13).

With [success/failure password databases](http://hello.k7mm.com/5i5.php?u=T2k4dmQybHJhVEl1Wkc5MlpXTnZkQzV2Y21jdlVHRnpjM2R2Y21SRVlYUmhZbUZ6WlE9PQ%3D%3D&b=13) (e.g. PAM) it's not possible to use non-plaintext mechanisms at all, because they only support verifying a known plaintext password.

Dovecot supports the following non-plaintext mechanisms:

* [CRAM-MD5](http://hello.k7mm.com/5i5.php?u=T2k4dmQybHJhVEl1Wkc5MlpXTnZkQzV2Y21jdlExSkJUUzFOUkRVPQ%3D%3D&b=13): Protects the password in transit against eavesdroppers. Somewhat good support in clients.
* [DIGEST-MD5](http://hello.k7mm.com/5i5.php?u=T2k4dmQybHJhVEl1Wkc5MlpXTnZkQzV2Y21jdlFYVjBhR1Z1ZEdsallYUnBiMjR2VFdWamFHRnVhWE50Y3k5RWFXZGxjM1JOUkRVPQ%3D%3D&b=13): Somewhat stronger cryptographically than CRAM-MD5, but clients rarely support it.
* SCRAM-SHA-1: Salted Challenge Response Authentication Mechanism (SCRAM) SASL and GSS-API Mechanisms. Intended as DIGEST-MD5 replacement.
* APOP: This is a POP3-specific authentication. Similar to CRAM-MD5, but requires storing password in plaintext.
* [NTLM](http://hello.k7mm.com/5i5.php?u=T2k4dmQybHJhVEl1Wkc5MlpXTnZkQzV2Y21jdlFYVjBhR1Z1ZEdsallYUnBiMjR2VFdWamFHRnVhWE50Y3k5T1ZFeE4%3D&b=13): Mechanism created by Microsoft and supported by their clients.
  + Optionally supported [using Samba's winbind](http://hello.k7mm.com/5i5.php?u=T2k4dmQybHJhVEl1Wkc5MlpXTnZkQzV2Y21jdlFYVjBhR1Z1ZEdsallYUnBiMjR2VFdWamFHRnVhWE50Y3k5WGFXNWlhVzVr&b=13).
* GSS-SPNEGO: A wrapper mechanism defined by [RFC 4178](http://hello.k7mm.com/5i5.php?u=T2k4dmRHOXZiSE11YVdWMFppNXZjbWN2YUhSdGJDOXlabU0wTVRjNA%3D%3D&b=13). Can be accessed via either GSSAPI or [Winbind](http://hello.k7mm.com/5i5.php?u=T2k4dmQybHJhVEl1Wkc5MlpXTnZkQzV2Y21jdlFYVjBhR1Z1ZEdsallYUnBiMjR2VFdWamFHRnVhWE50Y3k5WGFXNWlhVzVr&b=13).
* [GSSAPI](http://hello.k7mm.com/5i5.php?u=T2k4dmQybHJhVEl1Wkc5MlpXTnZkQzV2Y21jdlFYVjBhR1Z1ZEdsallYUnBiMjR2UzJWeVltVnliM009&b=13): Kerberos v5 support.
* RPA: Compuserve RPA authentication mechanism. Similar to DIGEST-MD5, but client support is rare.
* ANONYMOUS: Support for logging in anonymously. This may be useful if you're intending to provide publicly accessible IMAP archive.
* OTP and SKEY: One time password mechanisms.
* EXTERNAL: EXTERNAL SASL mechanism.

**Configuration**

By default only PLAIN mechanism is enabled. To use more, edit your /etc/dovecot/conf.d/10-auth.conf and set:

auth\_mechanisms = plain login cram-md5

实例：XMPP使用tls 和sasl登录

名词解释

TLS：安全传输层协议 TLS：Transport Layer Security  
名词：  
安全传输层协议（TLS）用于在两个通信应用程序之间提供保密性和数据完整性。该协议由两层组成： TLS 记录协议（TLS Record）和 TLS 握手协议（TLS Handshake）。较低的层为 TLS 记录协议，位于某个可靠的传输协议（例如 TCP）上面。

SASL全称Simple Authentication and Security Layer，是一种用来扩充C/S模式验证能力的机制。在Postfix可以利用SASL来判断用户是否有权使用转发服务，或是辨认谁在使用你的服务器。

-----------------------------------------------------------------

XMPP使用tls和sasl登录

步骤 1:客户端初始化流给服务器:

<stream:stream

      xmlns='jabber:client'

      xmlns:stream='http://etherx.jabber.org/streams'

      to='example.com'

      version='1.0'>

步骤 2:服务器向客户端发送流标签作为应答:

<stream:stream

      xmlns='jabber:client'

      xmlns:stream='http://etherx.jabber.org/streams'

      id='c2s\_234'

      from='example.com'

      version='1.0'>

步骤 3:服务器发送 STARTTLS范围给客户端（包括验证机制和任何其他流特性）:

<stream:features>

<starttls xmlns="urn:ietf:params:xml:ns: xmpp-tls"></starttls>

<mechanisms xmlns="urn:ietf:params:xml:ns:xmpp-sasl">

<mechanism>PLAIN</mechanism>

<mechanism>CRAM-MD5</mechanism>

<mechanism>ANONYMOUS</mechanism>

<mechanism>DIGEST-MD5</mechanism>

<mechanism>JIVE-SHAREDSECRET</mechanism>

</mechanisms>

<compression xmlns="http://jabber.org/features/compress">

<method>zlib</method>

</compression>

<auth xmlns="http://jabber.org/features/iq-auth"/>

<register xmlns="http://jabber.org/features/iq-register"/>

</stream:features>

说明：

1、 DIGEST-MD5：如果帐号和密码都在Client对象中提供了，这种机制是首选，即使没有TLS加密也是安全的。

2、 PLAIN：如果DIGEST-MD5无效，就使用此种机制。在没有TLS加密时是不安全的。

3、 ANONYMOUS：此种机制在没有提供帐号和密码时使用。服务器将随机产生临时帐号和资源，提供限制的有效服务。

4、 EXTERNAL：此种机制目前只对客户端提供了证书和保密关键字（private key）而有效，服务器试图通过外部计算出客户端。举例来说，使用提供的证书或IP地址

步骤 4:客户端发送 STARTTLS命令给服务器:

<starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>

步骤 5:服务器通知客户端可以继续进行:

<proceed xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>

Tls握手完成,进行SASL握手

步骤 6: TLS握手成功,客户端初始化一个新的流给服务器:

<stream:stream

      xmlns='jabber:client'

      xmlns:stream='http://etherx.jabber.org/streams'

      to='example.com'

      version='1.0'>

步骤 7:服务器通知客户端可用的验证机制:

<?xml version='1.0' encoding='UTF-8'?>

<stream:stream xmlns:stream="http://etherx.jabber.org/streams" xmlns="jabber:client" from="f3563766f3e3424" id="8af4a783" xml:lang="en" version="1.0">

<stream:features>

<mechanisms xmlns="urn:ietf:params:xml:ns:xmpp-sasl">

<mechanism>PLAIN</mechanism>

<mechanism>CRAM-MD5</mechanism>

<mechanism>ANONYMOUS</mechanism>

<mechanism>DIGEST-MD5</mechanism>

<mechanism>JIVE-SHAREDSECRET</mechanism>

</mechanisms>

<compression xmlns="http://jabber.org/features/compress">

<method>zlib</method>

</compression>

<auth xmlns="http://jabber.org/features/iq-auth"/>

<register xmlns="http://jabber.org/features/iq-register"/>

</stream:features>

步骤 8:客户端选择一个验证机制:

<auth xmlns='urn:ietf:params:xml:ns:xmpp-sasl' mechanism='DIGEST-MD5'/>

步骤 9:服务器发送一个 [BASE64]编码的挑战给客户端:

<challenge xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>

cmVhbG09InNvbWVyZWFsbSIsbm9uY2U9Ik9BNk1HOXRFUUdtMmhoIixxb3A9ImF1dGgi

  LGNoYXJzZXQ9dXRmLTgsYWxnb3JpdGhtPW1kNS1zZXNzCg==

  </challenge>

解码后的挑战信息是:

realm="somerealm",nonce="OA6MG9tEQGm2hh",

  qop="auth",charset=utf-8,algorithm=md5-sess

步骤 10:客户端发送一个[BASE64]编码的回应这个挑战:

<response xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>

  dXNlcm5hbWU9InNvbWVub2RlIixyZWFsbT0ic29tZXJlYWxtIixub25jZT0i

  T0E2TUc5dEVRR20yaGgiLGNub25jZT0iT0E2TUhYaDZWcVRyUmsiLG5jPTAw

  MDAwMDAxLHFvcD1hdXRoLGRpZ2VzdC11cmk9InhtcHAvZXhhbXBsZS5jb20i

  LHJlc3BvbnNlPWQzODhkYWQ5MGQ0YmJkNzYwYTE1MjMyMWYyMTQzYWY3LGNo

  YXJzZXQ9dXRmLTgK

  </response>

解码后的回应信息是:

username="somenode",realm="somerealm",

  nonce="OA6MG9tEQGm2hh",cnonce="OA6MHXh6VqTrRk",

  nc=00000001,qop=auth,digest-uri="xmpp/example.com",

  response=d388dad90d4bbd760a152321f2143af7,charset=utf-8

步骤 11:服务器发送另一个[BASE64]编码的挑战给客户端:

<challenge xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>

  cnNwYXV0aD1lYTQwZjYwMzM1YzQyN2I1NTI3Yjg0ZGJhYmNkZmZmZAo=

  </challenge>

步骤 12:客户端应答这个挑战:

<response xmlns='urn:ietf:params:xml:ns:xmpp-sasl'/>

步骤 13:服务器通知客户端验证成功:

<success xmlns='urn:ietf:params:xml:ns:xmpp-sasl'/>

以上只是讲了一个大致的流程，发送的每一个=属性到底是什么意思，以及用户名密码是怎么发出去的，没有讲得很详细，若要一个完整的DIGEST-MD5参考流程，可见下面的网站：

<http://wiki.xmpp.org/web/SASLandDIGEST-MD5>